INL/EXT-16-37953
Revision 0

RAVEN Beta 1.0 Release

Cristian Rabiti, Andrea Alfonsi, Joshua Cogliati, Diego Mandelli, Robert Kinoshita, Congjian Wang, Daniel Maljovec, Paul Talbot

February 2016

U.S. Department of Energy
Office of Nuclear Energy

INL/EXT-16-37953
Revision 0

RAVEN Beta 1.0 Release

Cristian Rabiti, Andrea Alfonsi, Joshua Cogliati, Diego Mandelli, Robert Kinoshita, Congjian Wang, Daniel Maljovec, Paul Talbot

February

Idaho National Laboratory
Nuclear Engineering Methods Development
Idaho Falls, Idaho 83415

http://www.inl.gov

Prepared for the
U.S. Department of Energy
Office of Nuclear Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

ABSTRACT

This report documents the release of the Risk Analysis Virtual Environment (RAVEN) code. A short description of the RAVEN code is provided, and the release process is described. The RAVEN code is a generic software framework to perform parametric and probabilistic analysis based on the response of complex system codes. RAVEN is capable of investigating the system response by sampling the input space using Monte Carlo, Grid, or Latin Hyper Cube sampling schemes and others, but its strength is focused toward system feature discovery, such as limit surfaces, which are hyper-surfaces separating regions of the input space leading to system failure, using dynamic supervised learning techniques. RAVEN has now increased in maturity enough for the Beta 1.0 release.

CONTENTS

FIGURES

TABLES

ACRONYMS

API	Application Programming Interfaces
CDF	Cumulative Distribution Function
DET	Dynamic Event Tree
LWRS	Light Water Reactor Sustainability
MC	Monte Carlo
MOOSE	Multiphysics Object-Oriented Simulation Environment
PDF	Probability Distribution Function
RAVEN	Risk Analysis Virtual Environment
RELAP-7	Reactor Excursion and Leak Analysis Program v.7
RISMC	Risk Informed Safety Margin Characterization
ROM	Reduced Order Model

1. Introduction

This report highlights the status of the RAVEN[1,2,3,4] code at the moment of the release of the beta1.0 version.

The current capabilities of the code are such it is deemed worthwhile to initiate a beta testing phase of the code. The software has already been licensed to fourteen institutions and provided to two other national laboratories as listed in Table 1.

Table 1: RAVEN Licensees

Argonne National Laboratory
Bechtel
Electrcite de France
FPoliSolutions, LLC
Idaho State University
Institute of Nuclear Safety System
Oak Ridge National Laboratory
Ohio State
Oregon State University
Politecnico Di Milano
Purdue University
Texas A&M
University of Idaho
University of Rome
University of Utah
Westinghouse

2. RAVEN Description

RAVEN is a generic software framework to perform parametric and probabilistic analysis based on the response of complex system codes. The initial development was aimed at providing dynamic risk analysis capabilities to the Reactor Excursion and Leak Analysis Program v.7[5] (RELAP-7) Thermal-Hydraulic code, currently under development at the Idaho National Laboratory (INL). The initial goal has been fully accomplished. RAVEN is now evolving toward a multi-purpose probabilistic and uncertainty quantification platform, capable to agnostically communicate with any external code (e.g. system simulators, etc.). The agnosticism is build using generic Application Programming Interfaces (APIs). These APIs are used to allow RAVEN to interact with any code as long as all the parameters that need to be perturbed are accessible through input files or via *Python* interfaces. RAVEN is capable of investigating the system response by sampling the input space using several sampling strategy such as Monte Carlo, Grid, or Latin Hyper Cube sampling schemes and many others. RAVEN's strength is focused toward system feature discovery, such as limit surfaces[6], which are hyper-surfaces separating regions of the input space leading to system failure, using dynamic supervised learning techniques. This section presents an overview of the software capabilities and their implementation schemes followed by some application examples.

2.1 RAVEN Initial Development

The development of RAVEN started in 2012 to satisfy the need to provide a modern risk evaluation framework. RAVEN's principal assignment is to provide the necessary software and algorithms in order to employ the concept developed by the Risk Informed Safety Margin Characterization (RISMC) program. RISMC is one of the pathways defined within the Light Water Reactor Sustainability (LWRS) program. In the RISMC approach, the goal is not just specifically identifying the frequency of an event potentially leading to a system failure, but also to analyze the "distance" and the drivers toward the

happening of key safety-related events. This approach may be used in identifying and increasing the safety margins related to those events. A safety margin is a numerical value quantifying the probability that a safety metric (e.g. as peak pressure in a pipe) is exceeded under certain conditions. The initial development of RAVEN has been focused on providing dynamic risk assessment capability to RELAP-7, currently under development at the INL. Most of the capabilities implemented for RELAP-7 are easily deployable for other system codes. For this reason, several side activities have been performed for coupling RAVEN with software such as BISON, RELAP5-3D, MELCOR, etc. The following sections are focused on the RAVEN software infrastructure and its functionality as present in the release beta 1.

3. Software Infrastructure

RAVEN has been developed in a highly modular and pluggable way in order to enable easy integration of different programming languages (i.e., C++, Python) and, as already mentioned, coupling with any system code. RAVEN is composed of three main software systems that can operate either in coupled or stand-alone mode:

- Control Logic System
- Graphical User Interface
- Probabilistic and Parametric framework

The control logic system and the Graphical User Interface are currently available for RELAP-7 only and distributed along with the RELAP-7 code. For this reason, attention for this release is focused on the probabilistic and parametric framework, which is what is commonly known as RAVEN.

3.1 Probabilistic and Parametric Framework

The probabilistic and parametric framework delivers the core of the RAVEN analysis capabilities. The main idea behind the design of the system is the creation of a multi-purpose framework with high flexibility with respect to the possible performable analysis. The framework must be capable of constructing the analysis/calculation flow at run-time, interpreting the user-defined instructions and assembling the different analysis tasks following a user specified scheme. In order to achieve such flexibility, combined with reasonably fast development, a programming language naturally suitable for this kind of approach was needed: *Python*.

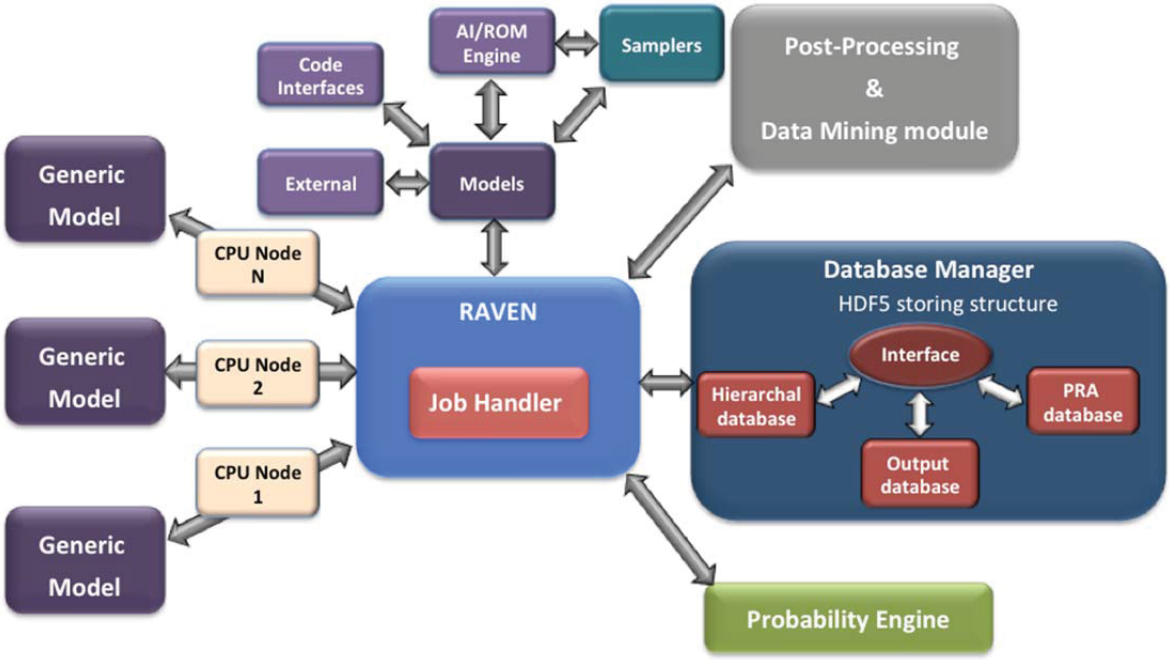

Figure 1: RAVEN framework layout

Hence, RAVEN is coded in *Python* and is characterized by an object-oriented design. The core of the analysis performable through RAVEN is represented by a set of basic components (objects) the user can combine, in order to create a custom analysis flow. A list of these components and a summary of their most important functionalities are reported as follows:

- *Distribution:* In order to explore the input/output space, RAVEN has the capability to perturb the input space (the initial conditions of a system code). The initial conditions, that represent the uncertain space, are generally characterized by probability distribution functions (PDFs), which need to be considered when the exploration of the system response is aimed to characterize the system behavior from a probabilistic point of view. For this purpose, a large library of PDFs is available.
- *Sampler:* A proper approach to sample the input space is fundamental for the optimization of the computational time. In RAVEN, a sampler determines a perturbation strategy that is applied to the input space of a system. The input space is defined through the connection of uncertain variables and their relative probability distributions.
- *Model:* A model is the representation of a physical system (e.g. Nuclear Power Plant); it is therefore capable of predicting the evolution of a system given a coordinate set in the input space. *Reduced Order Model (ROM)[7]:* The evaluation of the system response, as a function of the coordinates in the input space, is very computationally expensive, especially when brute-force approaches (e.g. Monte Carlo methods) are chosen as the sampling strategy. ROMs are used to lower this cost, reducing the number of needed points and prioritizing the area of the input space that needs to be explored. They can be considered as an artificial representation of the link between the input and output spaces for a particular system.

The list above is not comprehensive of all the RAVEN framework components (visualization, storage, post-processing infrastructures). Figure 1 shows a schematic representation of the whole framework, highlighting the communication pipes among the different modules and engines. In Figure 1

all the components listed above are schematically shown. In addition the data management, mining and processing modules are shown.

3.2 Probability Distributions

As already mentioned, the exploration of the input space, through the initial conditions (parameters) affected by uncertainties, needs to be performed using the proper probability density functions. RAVEN provides, through an interface to the BOOST library[8], the following univariate (truncated and not) distributions: Bernoulli, Binomial, Exponential, Logistic, Lognormal, Normal, Poisson, Triangular, Uniform, Weibull, Gamma, and Beta. The usage of univariate distributions for sampling initial conditions is based on the assumption that the uncertain parameters are not correlated with each other. Quite often uncertain parameters are subject to correlations and thus the univariate approach is not applicable. This happens when the value of multiple initial values are not independent but statistically correlated. RAVEN supports N-dimensional (N-D) PDFs. The user can provide the distribution values on either Cartesian or sparse grid, which determines the interpolation algorithm used in the evaluation of the imported CDF-PDF, respectively:

1. N-Dimensional Spline, for Cartesian grids
2. Inverse weight, for sparse grids

Internally, RAVEN provides the needed N-D differentiation or integration algorithms to compute the PDF from the CDF and vice versa. As already mentioned, the sampling methods use the distributions in order to perform probability-weighted sampling of the input space. For example, in the Monte Carlo approach, a random number [0,1] is generated (probability threshold) and the CDF, corresponding to that probability, is inverted in order to retrieve the parameter value to be used as coordinate in the input space simulation. The existence of the inverse for univariate distributions is guaranteed by the monotonicity of the CDF. For N-D distributions this condition is not sufficient since the $CDF(X) \rightarrow [0,1],\ X \in R^N$ and therefore it is not be a bijective function. From an application point of view, this means the inverse of a N-D CDF is not unique. As an example, Figure 2 shows a multivariate normal distribution for a pipe failure as function of the pressure and temperature. The plane identifies an iso-probability surface (in this case, a line) that represents a probability threshold of 50 percent in this example. Hence, the inverse of this CDF is an infinite number of points. As easily inferable, the standard sampling approach cannot directly be employed. When multivariate distributions are used, RAVEN implements a surface search algorithm for identifying the iso-probability surface location. Once the location of the surface has been found, RAVEN chooses, randomly, one point on it.

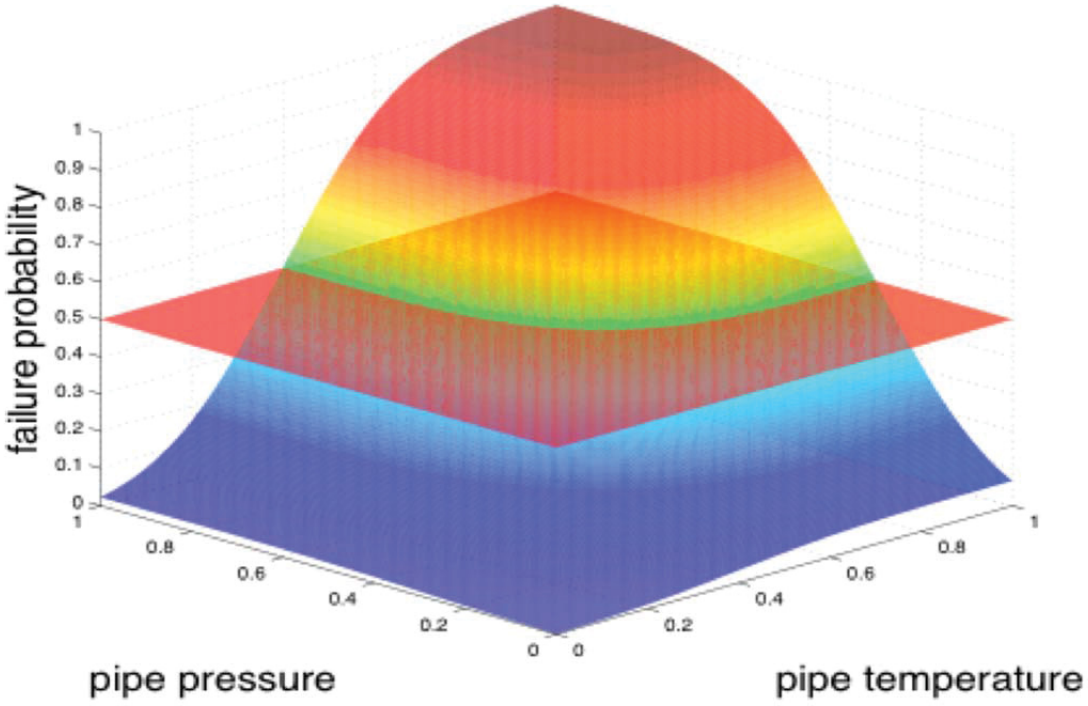

Figure 2: 2-D CDF, function of pressure and temperature

3.3 Sampler

The sampler is probably the most important entity in the RAVEN framework. It performs the driving of the specific sampling strategy and, hence, determines the effectiveness of the analysis, from both an accuracy and computational point of view. The samplers, that are available in RAVEN, can be categorized in three main classes:

- Forward
- Dynamic Event Tree (DET)[9]
- Adaptive[10]

The DET and Adaptive samplers are less common in literature. For this reason, they are going to be explained in more detail in the relevant sections.

3.3.1 Forward Samplers

The Forward sampler category includes all the strategies that perform the sampling of the input space without exploiting, through a dynamic learning approach, the information made available from the outcomes of calculation previously performed (adaptive sampling) and the common system evolution (patterns) that different sampled calculations can generate in the phase space (dynamic event tree). In the RAVEN framework, several different and well-known forward samplers are available:

- Monte Carlo (MC)
- Stratified based, whose most known version is the Latin Hyper-Cube Sampling (LHS)
- Grid Based
- Response Surface Design of Experiment

- Sparse Grid
- Factorials

3.3.2 Dynamic Event Tree Samplers

The Dynamic Event Tree methodologies are designed to take the timing of events explicitly into account, which can save a lot of computational time and handle particular type of phenomena that are intrinsically stochastic[11]. The main idea of this methodology is to let a system code determine the pathway of an accident scenario within a probabilistic environment. In this family of methods, a continuous monitoring of the system evolution in the phase space is needed. In order to use the DET-based methods, the generic driven code needs to have, at least, an internal trigger system and, consequently, a "restart" capability. In the RAVEN framework, 4 different DET samplers are available:

- Dynamic Event Tree (DET)
- Hybrid Dynamic Event Tree (HDET)
- Adaptive Dynamic Event Tree (ADET)
- Adaptive Hybrid Dynamic Event Tree (AHDET)

The ADET and the AHDET methodologies represent a hybrid between the DET/HDET and adaptive sampling approaches.

3.3.3 Adaptive Samplers

The Adaptive Samplers' family provides the possibility to perform smart sampling (also known as adaptive sampling) as an alternative to classical "Forward" techniques. The motivation is that system simulations are often computationally expensive, time-consuming, and high dimensional with respect to the number of input parameters. Thus, exploring the space of all possible simulation outcomes is infeasible using finite computing resources. During simulation-based probabilistic risk analysis, it is important to discover the relationship between a potentially large number of input parameters and the output of a simulation using as few simulation trials as possible.

Currently, RAVEN provides support for the following adaptive algorithms:

- Limit Surface Search[6]
- Adaptive Dynamic Event Tree[12]
- Adaptive Hybrid Dynamic Event Tree[13]
- Adaptive Sparse Grid[14]
- Adaptive Sobol Decomposition[15]

3.4 Models

The Model entity, in the RAVEN environment, is conceived as the "connecting pipeline" between the input and the output space. The RAVEN framework does not have any physical model (i.e. it does not posses the equations needed to simulate a generic physical system, such as Navier-Stokes equations, Maxwell equations, etc.), but implements Application Program Interfaces (APIs) by which any generic model can be integrated and interrogated. The RAVEN framework provides APIs for four different model categories: Codes, External Models, Post-Processors (PPs) and Reduced Order Models (ROMs). In the following paragraphs, a brief explanation of each of these Model categories is reported.

3.4.1 Code

The Code model is an implementation of the model API that allows communicating with external codes. The communication between RAVEN and any driven code is performed through the implementation of interfaces directly operated by the RAVEN framework.

The procedure of coupling a new code/application with RAVEN is a straightforward process. The coupling is performed through a Python interface that interprets the information coming from RAVEN and translates them to the input of the driven code. The coupling procedure does not require modifying

RAVEN itself. Instead, the developer creates a new Python interface that is going to be embedded in RAVEN at run-time (no need to introduce hard-coded coupling statements). This interface needs to be placed in a folder (whatever name) located in (see Figure 3) ~/raven/distribution/raven/framework/CodeInterfaces/:

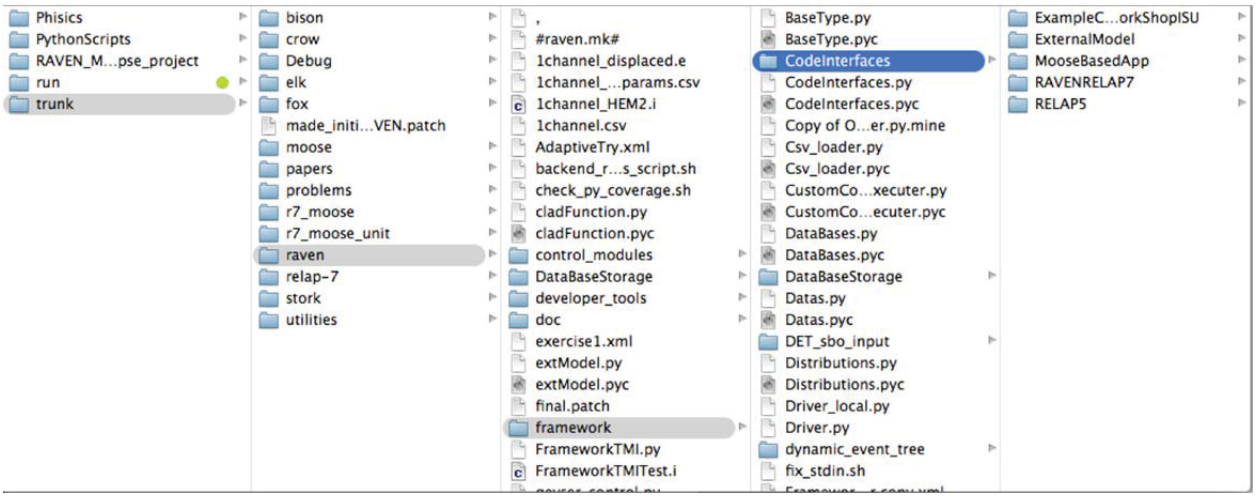

Figure 3: Code Interface Location

At the initialization stage, RAVEN imports all the Interfaces that are contained in this directory and performs some preliminary cross-checks.

If the coupled code is parallelized and/or multi-threaded, RAVEN is going to manage the system in order to optimize the computational resources in both workstations and High Performance Computing systems.

Currently, RAVEN has model API implementation for RELAP5-3D, RELAP-7, any MOOSE-based application, SASS and Modelica.

3.4.2 External Model

The External model allows the user to create, in a Python file (imported, at run-time, in the RAVEN framework), its own model (e.g. set of equations representing a physical model, connection to another code, control logic, etc.). This model will be interpreted/used by the framework and, at run-time, will be directly used by RAVEN.

3.4.3 Post-Processor

The Post-Processor model represents the container of all the post-processing capabilities in the RAVEN code. This model is aimed to process the data (for example, derived from the sampling of a physical code) in order to identify representative Figure of Merits. This system has been designed and, presently, is under heavy development. Currently, the following post-processors are available:

- *Basic Statistics*, container of the algorithms to compute many of the most important statistical quantities. This post-processor is able to compute mean, sigma/variance, variation coefficient, skewness, kurtosis, median, percentiles and all the principal matrix quantities such as covariance, sensitivity (either least-squared or variance weighted) and correlation matrices;
- *Comparison Statistics*, aimed to employ validation and verification metrics;
- Limit Surface, aimed to compute the limit surface in the input space (i.e. the hyper-surface that represents the boundary between the failure/success of the system);
- *Limit Surface Integral*, intended to compute the integral of the limit surface that corresponds to the probability of failure;

- *Safest Point*, provides the coordinates of the farthest point from the limit surface that is given as an input. The safest point coordinates are expected values of the coordinates of the farthest points from the limit surface in the space of the "controllable" variables based on the probability distributions of the "non-controllable" variables. The term "controllable" identifies those variables that are under control during the system operation, while the "non-controllable" variables are stochastic parameters affecting the system behavior randomly;
- *External Post-Processor*, user-defined post-processor;
- *Topological Decomposition*, aimed to compute an approximated hierarchical Morse-Smale complex which will add two columns to a data-set, performing a topological decomposition of such data-set;
- *Data Mining*, container of all the RAVEN data mining, clustering and dimensionality reduction techniques aimed to identify dominant and common patterns in high-dimensionality data.

3.4.4 Reduced Order Model

As briefly mentioned, a ROM is a mathematical representation of a system, used to predict a selected output space of a physical system. The "training" is a process that uses sampling of the physical model to improve the prediction capability (capability to predict the status of the system given a realization of the input space) of the ROM. More specifically, in RAVEN the Reduced Order Model is trained to emulate a high fidelity numerical representation (system codes) of the physical system. Two general characteristics of these models can be generally assumed (even if exceptions are possible):

1. The higher the number of realizations in the training sets, the higher is the accuracy of the prediction performed by the reduced order model. This statement is true for most of the cases although some ROMs might be subject to the over-fitting issues. The over-fitting phenomenon is not discussed here, since its occurrence is highly dependent on the algorithm type, (and there is a large number of ROM options available in RAVEN). Every time the user chooses a particular reduced order model algorithm to use, the user should consult the relevant literature;
2. The smaller the size of the input domain with respect to the variability of the system response, the more likely the surrogate model will be able to represent the system output space.

Within the RAVEN framework several ROM types are available, most of them are inherited from scikit-learn[16], and others are developed by the RAVEN team, such as the ones based on stochastic polynomials[17].

3.5 Simulation Environment

RAVEN can be considered as a pool of tools and data. Any action in which the tools are applied to the data is considered a calculation "step" in the RAVEN environment. Simplistically, a "step" can be seen as a transfer function between the input and output space through a Model (e.g. Code, External, ROM or Post-Processor). One of the most important types of step in the RAVEN framework is called "multi-run", that is aimed to handle calculations that involve multiple runs of a driven code (sampling strategies). Firstly, the RAVEN input file associates the variables to a set of PDFs and to a sampling strategy. The "multi-run" step is used to perform several runs in a block of a model (e.g. in a MC sampling).

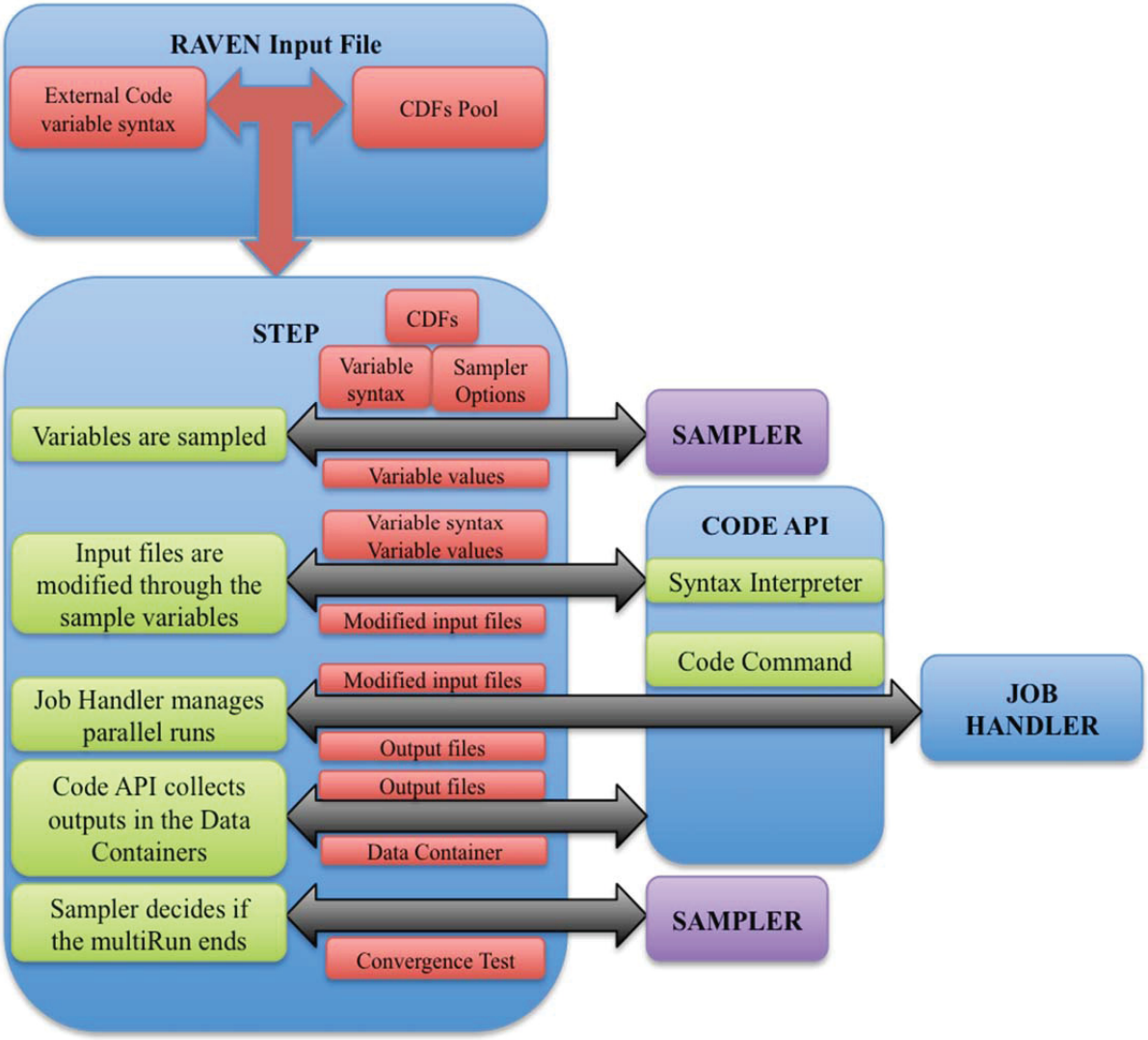

Figure 4: Calculation flow for a multi-run sampling

At the beginning of each sub sequential run, the sampler provides the new values of the variables to be explored. The code API places those values in the input file, if an external code is used. At this point, the code API generates the run command and asks to be queued by the job handler. The job handler manages the parallel execution of as many runs as possible within a user prescribed range and communicates with the step controller when a new set of output files are ready to be processed. The code API receives the new input files and collects the data in the RAVEN internal format. The sampler is queried to assess if the sequence of runs is ended, if not, the step controller asks for a new set of values from the sampler and the sequence is restarted as described in Figure 4. The job handler is currently capable to run different run instances of the code in parallel and can also handle codes that are multi-threaded or using any form of parallel implementation. RAVEN also has the capability to plot the simulation outcomes while the set of sampling is performed and to store the data for later recovery.

4. Release Methods

Several methods have been created for obtaining RAVEN. For developers as well as users who want the most up to date version of RAVEN, the gitlab[18] software can be used. This software provides configuration management for the source code and tracking of the versions. For other users the RAVEN

team periodically creates prepackaged versions of the RAVEN software. For each version of the code that is released, a source code package which can be used on Linux (Ubuntu and Fedora are regularly tested, others including OpenSUSE and SUSE Enterprise have been successfully used) and Mac OSX (Yosemite and El Capitan) is created. A precompiled binary package is created for the two newest versions of Mac OSX (Yosemite and El Capitan). There is also a binary package that can be used to install the libraries that RAVEN depends on in Mac OSX.

5. Quality Assurance Practices

The Quality Assurance practices adopted for the development of the code are described in the INL plans[19,20,21,22,23].

While the above described documents describe the development cycle and therefore control the code available directly from the gitLab server, periodically more extensive tests are automatically run on several different Linux virtual machines with different libraries. Those tests check for compatibility problems before the distribution of the code packages. For released packages, the packages are first released to the developers and then the developers test them over a variety of systems. This allows additional installation and compatibility problems to be found.

6. Other Documentation

One additional document is attached to this report "RAVEN User Manual". This is the official RAVEN manual updated up to the beta 1 version of the code. This document could be retrieved in the software repository (GitLab) and is part of the distribution packages.

7. Obtaining Software

The first step to obtaining RAVEN is to obtain a license for the software. Contact the raven service id email RAVEN@inl.gov to begin the process. Once the license has been obtained, the released version can provided.

As a part of the license process INL asks the requesting organization to fill a form, the Program Participation Form, where the people that would like to receive directly a version of the code from INL are listed. Once a license of the code is obtained there are two ways to get access to the RAVEN Software.

7.1 GitLab Repository Access

Once the license has been released any person in the Program Participation Form can ask to have access directly to the GitLab repository of the code. This way of obtaining the code is preferred for advanced users that want always the latest version of the code, for developers, and for people that have a close cooperation with the development team to ensure an effective and fast channel to address issues arising from usage experience. This is of course the preferred way to distribute the code to beta testers to have a fast feedback time with the testers.

To get access to the GitLab repository, first the user needs an account at the INL HPC environment. The first step in getting access to the INL HPC environment is going to: https://secure.inl.gov/caims/, choose HPC as the application and fill out the form listing one of the following people as the INL contact:

Joshua Cogliati (joshua.cogliati@inl.gov)

Cristian Rabiti (cristian.rabiti@inl.gov)

Andrea Alfonsi (andrea.alfonsi@inl.gov)

Once the account has been approved (for non us citizens this might take some extra time since it requires a background check), the user needs to perform an access to: https://hpcgitlab.inl.gov/. Only after the first access has been performed can the INL sponsor grant access to the RAVEN code.

The link http://mooseframework.org/wiki/hpcgitlabconnectivity/ describes how to access hpcgitlab from outside of INL. For accessing the gitlab versions, ssh keys will need to be setup so that gitlab can be used with ssh. Further detail on ssh keys setup is available in the gitlab help: https://hpcgitlab.inl.gov/help/ssh/README.

Once access has been gained to the hpcgitlab it is possible also to download some of the release packages:

- Miniconda libraries contain the libraries needed for running RAVEN on OSX: https://hpcgitlab.inl.gov/idaholab/raven/wikis/binary/raven_miniconda.dmg
- Complete install contains both the libraries and RAVEN and is for OSX: https://hpcgitlab.inl.gov/idaholab/raven/wikis/binary/raven_framework_complete_miniconda_OSX_10.10.5_release_beta_1.0.dmg
- Source package contains the source code for RAVEN and portions of MOOSE: https://hpcgitlab.inl.gov/idaholab/raven/wikis/binary/raven_framework_release_beta_1.0_source.tar.gz

7.2 Accessing the RAVEN packages

If GitLab access is not used, once the license has been obtained, the user should email RAVEN@inl.gov with a request for the current release packages and an email will be sent that contains a link to download the software packages.

8. Installing RAVEN

Once RAVEN has been obtained, the RAVEN User Manual[24] documents how to install both the dependencies that RAVEN needs to run, and RAVEN itself. The dependency installation is documented in Section 4: RAVEN Dependencies Installation, and the installation of RAVEN is document in Section 5: RAVEN Installation in the RAVEN User Manual.

9. REFERENCES

1 A. Alfonsi, C. Rabiti, D. Mandelli, J. Cogliati, and R. Kinoshita, "Raven as a tool for dynamic probabilistic risk assessment: Software overview," in Proceeding of M&C2013 International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering, on CD-ROM, May 5-9, Sun Valley (2013).

2 C. Rabiti, A. Alfonsi, D. Mandelli, J. Cogliati, R. Martinueau, C. Smith, "Deployment and Overview of RAVEN Capabilities for a Probabilistic Risk Assessment Demo for a PWR Station Blackout," Idaho National Laboratory report: INL/EXT-13-29510 (2013).

3 A. Alfonsi, C. Rabiti, D. Mandelli, J. Cogliati, R. Kinoshita, and A. Naviglio, "RAVEN and dynamic probabilistic risk assessment: Software overview," in Proceedings of ESREL European Safety and Reliability Conference, Wroclaw, Poland (2014).

4 A. Alfonsi, C. Rabiti, D. Mandelli, J. Cogliati, R. Kinoshita, "Performing Probabilistic Risk Assessment Through RAVEN", in Proceedings of American Nuclear Society 2013 Annual Meeting "Next Generation Nuclear Energy: Prospects and Challenges", Atlanta, GA (2013).

5 D. Anders, R. Berry, D. Gaston, R. Martineau, J. Peterson, H. Zhang, H. Zhao, L. Zou, "RELAP-7 Level 2 Milestone Report: Demonstration of a Steady State Single Phase PWR Simulation with RELAP-7," Idaho National Laboratory: INL/EXT-12-25924 (2012).

6 A. Alfonsi, C. Rabiti, D. Mandelli, J. Cogliati, S. Sen, C. Smith,"Improving Limit Surface Search Algorithms in RAVEN Using Acceleration Schemes", INL/EXT-15-36100, 2015

7 D. Mandelli, C. L. Smith, A. Alfonsi, C. Rabiti, J. Cogliati, P. W. Talbot, I. Rinaldi, D. Maljovec, B. Wang, V. Pascucci, H. Zhao, "Reduced Order Model Implementation in the Risk-Informed Safety Margin Characterization Toolkit", INL/EXT-15-36649, 2015

8 Boost C++ Libraries, http://www.boost.org/, Retrieved 2016-Feb-22

9 A. Alfonsi, C. Rabiti, D. Mandelli, J. Cogliati, R. Kinoshita, and A. Naviglio, "Dynamic event tree analysis through Raven," in Proceedings of ANS PSA 2013 International Topical Meeting on Probabilistic Safety Assessment and Analysis, 2013.

10 D. Mandelli, C. L. Smith, A. Alfonsi, C. Rabiti, J. Cogliati, "Improved Sampling Algorithms in the Risk-Informed Safety Margin Characterization Toolkit", INL/EXT-15-35933, 2015

11 C. Rabiti,, D. Mandelli, A. Alfonsi, J. Cogliati, and B. Kinoshita, "Mathematical framework for the analysis of dynamic stochastic systems with the raven code," in Proceedings of International Conference of Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2013), Sun Valley, ID, pp. 320–332, 2013.

12 A. Alfonsi, C. Rabiti, D. Mandelli, J. Cogliati, R. Kinoshita, "Adaptive Dynamic Event Tree in RAVEN code" in Proceedings of the ANS Winter Meeting, Anaheim, CA, November 2014.

13 A. Alfonsi, C. Rabiti, D. Mandelli, S. Sen, J. Cogliati, "Dynamic Event Tree advancements and control logic improvements", INL/EXT-15-36758, 2015.

14 T. Gerstner and M. Griebel. Dimension-adaptive tensor-product quadrature. Computing, 71(1):65-87, 2003.

15 D. Ayres, M.D. Eaton, "Uncertainty quantification in nuclear criticality modelling using a high dimensional model representation", Annals of Nuclear Energy, Volume 80, June 2015, Pages 379-402, ISSN 0306-4549, http://dx.doi.org/10.1016/j.anucene.2015.02.024.

16 Pedregosa et al., "Scikit-learn: Machine Learning in Python", , JMLR 12, pp. 2825-2830, 2011.

17 C. Rabiti, P. Talbot, A. Alfonsi, D. Mandelli, J. Cogliati, "Implementation of Stochastic Polynomials Approach in the RAVEN Code", INL/EXT-13-30611, 2013.

18 https://about.gitlab.com Code, test, and deploy together with GitLab open source git repo management software, Retrieved 2016-Februrary-22

19 R. Martineau,, "Maintenance and Operations Plan for the MOOSE Project PLN-4003", INL Document, 2015

20 R. Martineau, "Configuration Management Plan for Modeling and Simulation Software PLN-4004", INL Document, 2015

21 R. Martineau, "Software Quality Assurance Plan for Modeling and Simulation Software PLN-4005", INL Document, 2015

22 R. Martineau,, "Verification & Validation Plan for Modeling and Simulation Software PLN-4006", INL Document, 2015

23 R. Martineau, , “Project Management Plan for Modeling and Simulation Software PLN-4213”, INL Document, 2015

24 C. Rabiti, A. Alfonsi, J. Cogliati, D. Mandelli, C. Wang, R. Kinoshita, S. Sen, “RAVEN User Manual”, INL/EXT-15-34123, Revision 4, 2016.